Assignment 2: Coding Basics

Laila Abed

OVERVIEW

This exercise accompanies the lessons/labs in Environmental Data Analytics on coding basics.

Directions

1. Rename this file `<FirstLast>_A02_CodingBasics.Rmd` (replacing `<FirstLast>` with your first and last name).
2. Change "Student Name" on line 3 (above) with your name.
3. Work through the steps, **creating code and output** that fulfill each instruction.
4. Be sure to **answer the questions** in this assignment document.
5. When you have completed the assignment, **Knit** the text and code into a single PDF file.
6. After Knitting, submit the completed exercise (PDF file) to Canvas.

Basics, Part 1

1. Generate a sequence of numbers from one to 55, increasing by fives. Assign this sequence a name.
2. Compute the mean and median of this sequence.
3. Ask R to determine whether the mean is greater than the median.
4. Insert comments in your code to describe what you are doing.

```
#1.
Sequence_int <- seq(1, 55, 5) # create a sequence "Sequence _int" and
#assigning sequential values starting from 1 to 55 increasing by 5.
Sequence_int # show the values of Sequence_int.
```

```
##  [1]  1  6 11 16 21 26 31 36 41 46 51
```

```
#2.
Mean_Seq <- mean(Sequence_int) # calculate mean and assign the value to Mean_seq
Mean_Seq # show the value
```

```
## [1] 26
```

```
Median_Seq <- median (Sequence_int) # calculate median and assign value to Median_seq
Median_Seq # show the value
```

```
## [1] 26
```

```r
#3.
 Mean_Seq > Median_Seq # compare mean and median give true if mean greater
```

```
## [1] FALSE
```

```r
 #than median and false otherwise.
```

Basics, Part 2

5. Create three vectors, each with four components, consisting of (a) student names, (b) test scores, and (c) whether they are on scholarship or not (TRUE or FALSE).
6. Label each vector with a comment on what type of vector it is.
7. Combine each of the vectors into a data frame. Assign the data frame an informative name.
8. Label the columns of your data frame with informative titles.

```r
#5. #6.
student_names <- c("Ali","Sidd","Noor","Kelly") #student_names vector class is character
test_scores <- c(90, 95, 80,98) #test_scores vector class is numeric
scholarship <- c(TRUE, FALSE, TRUE, TRUE) #scholarship vector class is logic
#7.
data_frame_college <- data.frame(student_names, test_scores, scholarship)
#create data frame from the three vectors
data_frame_college
```

```
##   student_names test_scores scholarship
## 1           Ali          90        TRUE
## 2          Sidd          95       FALSE
## 3          Noor          80        TRUE
## 4         Kelly          98        TRUE
```

```r
#8.
names(data_frame_college) <- c("Student Name","Test Score","Has Scholraship")
#naming the columns in the data frame
# View(data_frame_college)
# Another way to create the data frame and change the names of the columns
data_frame_college2 <- data.frame(
  StudentName = student_names,
  TestScore = test_scores,
  HasScholarship = scholarship
)
data_frame_college2
```

```
##   StudentName TestScore HasScholarship
## 1         Ali        90           TRUE
## 2        Sidd        95          FALSE
## 3        Noor        80           TRUE
## 4       Kelly        98           TRUE
```

9. QUESTION: How is this data frame different from a matrix?

Answer: - Data frame is a list of vectors (columns) of equal length, it can have different types of data in each column/vector. For example three columns, each has different type of data numeric, character, logical. Columns have names. If you know the name of the column, you can get the data under this column name. - Matrix is two dimensions array used widely in mathematics, it must have the same type of data (all elements from the smae type). For example all numeric, all character.Columns and rows don't have names. You have to enter row number and column number to get the data.

10. Create a function with one input. In this function, use `if...else` to evaluate the value of the input: if it is greater than 50, print the word "Pass"; otherwise print the word "Fail".
11. Create a second function that does the exact same thing as the previous one but uses `ifelse()` instead if `if...else`.
12. Run both functions using the value 52.5 as the input
13. Run both functions using the **vector** of student test scores you created as the input. (Only one will work properly...)

```
#10. Create a function using if...else
function_pass_fail <- function (x){
  if (x>50) {print ("Pass") }
  else {print ("Fail") }
}


#11. Create a function using ifelse()

function_evaluate_pass_fail <- function(y){

  Value <- ifelse(y > 50, "Pass", "Fail")
  print(Value)
  }
#If X > 50, if TRUE Print Pass, if FALSE Print Fail.


#12a. Run the first function with the value 52.5
function_pass_fail (52.5)
```

```
## [1] "Pass"
```

```
#12b. Run the second function with the value 52.5
function_evaluate_pass_fail (52.5)
```

```
## [1] "Pass"
```

```
#13a. Run the first function with the vector of test scores
results <- c(100, 30, 85, 45, 90)
#function_pass_fail (results) # 'If'... 'else' did not work because the function
#expects a single numeric value, when we give it vector of several values won't work.

#13b. Run the second function with the vector of test scores
function_evaluate_pass_fail (results)   # 'ifelse' worked well, it can deal with
```

```
## [1] "Pass" "Fail" "Pass" "Fail" "Pass"
```

```
#vectors.
```

14. QUESTION: Which option of `if...else` vs. `ifelse` worked? Why? (Hint: search the web for "R vectorization")

    Answer: 'If' ... 'else' did not work because the function expects a single numeric value, when we give it a vector of several values won't work. While 'ifelse' worked well, it can deal with vectors. It applies the logic (condition) to each value and return the result.

NOTE Before knitting, you'll need to comment out the call to the function in Q13 that does not work. (A document can't knit if the code it contains causes an error!)